3D-FEATURE RECOGNITION FROM MEASURED DATA

M. Janssens[1], W. van Wijck[2], N. D. du Preez[2]

ABSTRACT

This paper presents a method to automatically extract analytical entities like planes, spheres and cylinders from a file containing a cloud of points. The method facilitates the manipulation and reduction of large data sets and the evaluation of it. It can be used as a design tool, a quality control tool, data-processing tool or a data reduction tool. From a database of points, the user can automatically extract a subset of points belonging to an analytical entity of interest, within a predefined but adjustable level of confidence. If necessary, the dimensional parameters of the entity can also be calculated.

The method is based on the subtle statistical properties of the least-squares technique that makes it compliant with the strict regulations in the co-ordinate measuring arena. Its robustness guarantees the applicability to less accurate environments than precision engineering.

OPSOMMING

Hierdie artikel bespreek 'n metode met behulp waarvan analitiese voorwerpe soos vlakke, sfere en silinders outomaties vanuit 'n wolk van datapunte, onttrek kan word. Die metode is geskik vir die hantering, manipulasie, reduksie en evaluasie van groot data-stelle. Dit kan gebruik word as 'n gereedskapstuk vir ontwerp, gehaltebeheer, dataverwerking en data-reduksie. Gegewe 'n databasis van punte, kan die gebruiker die subset van punte wat tot enige analitiese voorwerp van belang behoort, outomaties binne 'n voorafgespesifiseerde, maar verstelbare, vlak van vertroue onttrek. Hierbenewens en indien nodig, kan die dimensionele parameters en afmetings van die betrokke entiteit ook bereken word.

Die algoritme maak van die kleinste-kwadrate metode gebruik, sodat elke passings-parameter statisties kwantifiseerbaar en verantwoordbaar is. In hierdie opsig voldoen dit aan die streng regulasies wat die koördinaat-meet arena kenmerk. Die robuustheid van die metode, maak dit ook geskik vir toepassing in minder akkurate omgewings as presisie-ingenieurswese.

1. INTRODUCTION

The development of the method discussed in this paper was sparked by the need for a design tool for rapid prototyping. STL is by far the most widely used standard for data storage and transmission in rapid prototyping. STL is a triangle-based representation for object data. Practically, this may be thought of as a cloud of points with a topological structure. During the design of such an STL-based system, one often needs to manipulate analytical entities that are represented as points. The feature recognition system was developed to enable the selection of analytical entities automatically.

Another field of application is metrology. Modern measuring techniques allow very fast measuring. Unfortunately, this usually involves an enormous and often unmanageable amount of data. Extracting analytical entities often leads to a considerable reduction of the data set. The statistical correctness of the technique guarantees compliance with the relevant standards.

[1] Materialise N.V., Technologielaan 15, 3001 Leuven, Belgium
[2] Department of Industrial Engineering, University of Stellenbosch

The first section gives a short description of the least-squares method, emphasising the statistical aspect. The incremental range method described in the second section relies heavily on the statistical properties of the least-squares method. Later in the second section, the exact implementation of the method is described. The examples clearly show the working of the method. The article ends with two ideas for further development.

2. LEAST-SQUARES REVISITED

Least-squares techniques are perhaps the most frequently applied mathematical tools in engineering. Nevertheless, this section briefly reiterates some of the basic but important properties of the method of least-squares fitting. The method of least-squares goes back to C. F. Gauss (1777-1855), who used it to compute the trajectory of the planets around the sun from measured data. He obtained results that were far more accurate than the results obtained by "classical" methods. Nowadays, the least-squares method is considered classical. This section explains the statistical basis of the least-squares method in the context of 3D-feature recognition using a cylindrical object as an example.

The input for the example problem is a set of measured points $\{p_i\}$. A least-squares cylinder over a set of points is a cylinder such that the function F_{Cyl}, defined over all cylinders C as:

$$F_{Cyl}(C) = \sum_{i=1}^{n} d(C, p_i)^2 \qquad (1)$$

where *d is* the distance function,

has a minimal value. There are only two practical cases where this problem does not have a unique solution. The first case arises when there are fewer than five points in the point set, in which case an infinite number of least-squares cylinders exist. The second case arises when there are five or more points but all points are in one plane. Generally, no solution exists in the latter case. Although it is possible to construct point sets that have a finite number of solutions, such sets never occur in practice and therefore are unimportant. Keeping the above remarks in mind, it is safe to talk about ***the*** least-squares cylinder of a set of points. Finding the least-squares cylinder involves solving an optimisation problem. Forbes [2] is a standard work on finding least-squares geometrical entities (planes, spheres, cylinders, etc.). The method that was used to generate the examples in this paper is relatively fast. A detailed discussion of this method can be found in Janssens [4].

Even if all the points in a practical point set do describe and belong to a specific cylinder, these points will not exactly coincide with the (generally unknown) nominal cylinder. The points will deviate from the nominal cylinder according to some statistical distribution having a mean close to zero and a variance which may be calculated as:

$$\sigma^2 = \frac{\sum_{i=1}^{n} d(C, p_i)^2}{n} \qquad (2)$$

Because the above deviations are the result of several small contributing factors, they are likely to be normally distributed according to the central limit theorem.

When the nominal cylinder is unknown then statistically speaking, the least-squares cylinder represents the best "average" cylinder in the sense that it will minimise equation (1). For the least-square cylinder, the sample variance can be calculated using the following formula:

$$s^2 = \frac{\sum_{i=1}^{n} d(C, p_i)^2}{n - 5} \qquad (3)$$

Notice that equation (3) requires a correction for the number of degrees of freedom which in this case is five less than the actual number of data points. This is because a minimum of five points is necessary to define the least-square cylinder. The correction is especially important if the number of data points is small. When a large number of data points are available the correction becomes unimportant.

3. INCREMENTAL RANGE CHECKING

Useful information about an entity and its direct environment is contained within the least-squares statistics and in particular the resulting deviation. The standard deviation describes the distribution of the distances from the points in the data set to the entity (e.g. cylinder). A simple statistical acceptance test, based on these distances, can therefore be performed to determine whether a candidate-point belongs to the cylinder or not.

The aforementioned acceptance test can be formulated in two ways. The first and statistically more correct approach would be to use the resulting standard deviation of the least-squares cylinder. As an alternative to this, a pre-defined standard deviation can be used. The latter approach is more robust and often more useful to the user.

Both acceptance tests lead to iterative algorithms that give very good results. Figure 1 shows the flowchart. The algorithm starts by manually selecting some points on the cylinder. If the former acceptance test is used, six or more points are necessary, otherwise the standard deviation will be undefined. For the latter acceptance test, a minimum of five starting points is necessary. In the second step, the least-squares cylinder is computed from the selected points. The third step adds all points in the cloud that pass the acceptance test. Steps two and three are repeated until no new points are added. The set of selected points will grow with every iteration until all points belonging to the cylinder have been identified and selected. The last computed least-squares cylinder is the one of interest to the user.

Both acceptance tests imply that the points that are used for computing the least-squares cylinder, are indeed part of the cylinder. This means that the algorithm does not work when the initially selected points are not on a cylinder. In this case, two things can happen: either the selected area will remain very small or the least-squares fitting algorithm will not succeed in finding a cylinder. Because both these cases can be detected quite easily, the error type can be identified and the user notified.

One of the main drawbacks of the algorithm is its (lack of) speed which is primarily determined by the least-squares procedure. At the beginning of the recognition process, only a few points are selected on a small part of the cylinder. This leads to a very ill conditioned least-squares equation, which in turn leads to a slow solution. As the recognition process progresses, the selected area of the cylinder increases, with an accompanying increase in the number of points for the least-squares equation. The speed of the algorithm however remains poor because any improvement in the condition of the least-squares equation is neutralised by the growing number of points that must be processed.

The least-squares fitting algorithm for cylinders (as for most other entities) is an iterative algorithm that requires a starting value. By using the value of the previous iteration as the starting value for the next, the recognising algorithm is greatly optimised. Apart from this, the only way to speed up the algorithm is to increase the speed of the fitting algorithm itself.

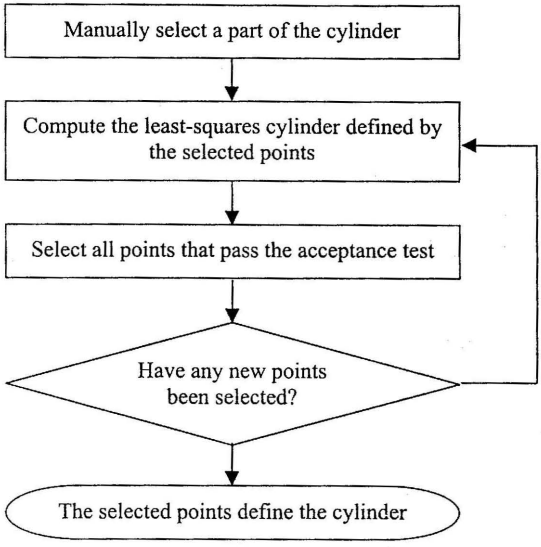

Figure 1: Flowchart of the recognising algorithm

4. STATISTICAL PITFALL

The recognising algorithm is extremely simple and the theory behind it particularly suitable for the specific application. It can be easily extended to include other entities than cylinders. Unfortunately, an important pitfall can occur in practical examples. In most cases, the entity to be recognised does not end abruptly but instead smoothly blends into an adjacent entity. The transition zone, while not part of the entity, will also be selected with the entity if the transition is very gradual. The second example below illustrates this.

The problem will almost never occur when using the first acceptance test, because in this test the standard deviation is very strict. However in the second acceptance test where a predefined deviation is used, the user must select one that is very close to the actual standard deviation, else non-representative points will be included in the selection. This of cause assumes that the user will have some idea about the size of the true deviation. Fortunately, in practice this error has a small effect on the parameter values of the least-squares entity, because the proportion of non-representative points is normally small. It does however effect the selected zone.

There are basically two ways of avoiding this pitfall. One is by checking after each step if there was a "significant" change in the calculated standard deviation and in the parameter values of the entity. This method borrows from the Kalman Filter. De Geeter [1] discusses this matter. However, the method's complexity only allows very simple entities like planes to be recognised.

The second method registers and functionally relates the size of the deviation with its physical position on the entity. This is in fact only a minor change of the previous method. Instead of grouping the points in terms of *when* they are added to the selection, they are grouped according to *where* they occur on the entity.

5. EXAMPLES

The two examples presented in this section illustrate the method as it is implemented and integrated in the Magics software, which is a product of Materialise N.V. for rapid prototyping. Both examples deal with measured and therefore inaccurate data. The same data set is used in both cases. Figure 2 shows a shaded image of the data set. The data are in STL-format, so triangular information is available.

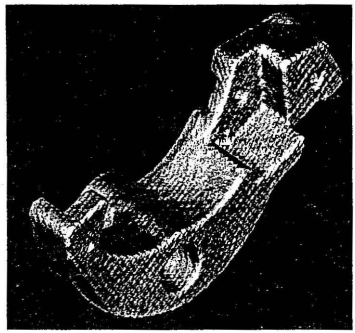

Figure 2: The data set

The first example shows the recognition of the bottom plane. Initially, three points are selected by manually selecting one triangle on the screen. (Note that in contrast to a cylinder, only three points are necessary to define a plane.) After one step, a large area of the plane is already selected. The selected area is shown in Figure 3. It is clear that the initially computed plane (using the manually selected three points as starting values) does not completely represent the bottom plane.

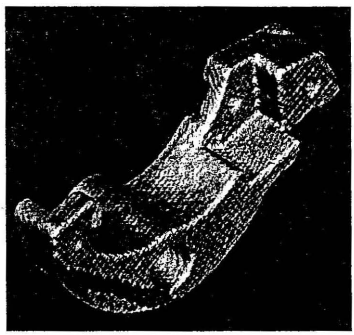

Figure 3: The selected points after one step

After the second iteration, the least-squares plane computed using the newly selected points represents a better approximation. Now more points are added (see Figure 4), but still not all points belonging to the plane are selected.

After the third step, no more points are added. The incremental algorithm stops and the resulting least-squares plane represents the end-result. The final plane is shown in Figure 5. The statistical pitfall referred to in the previous section does not create a problem here, because the transition to the adjacent entities is sharp. The points belonging to the transition zone defines an "edge". These points are immediately out of the expected range and consequently do not pass the acceptance test.

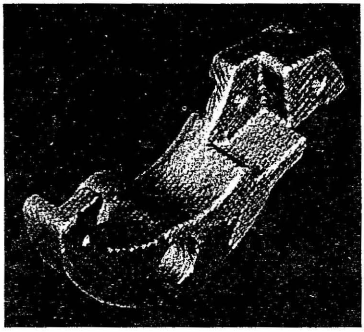

Figure 4: The selected points after two steps

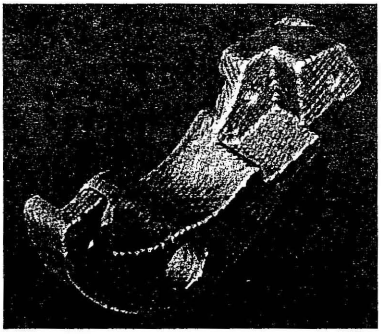

Figure 5: The resulting least-squares plane

The second example illustrates the recognition of a cylinder. The strategy is exactly the same, except for the starting selection. First, a triangle is selected manually. However, this creates a problem because three points are not sufficient to define a (least-squares) cylinder. The present strategy of overcoming this problem is rather opportunistic and involves the manual selection of at least two or three more neighbouring triangles of the initially selected triangle. An implicit assumption is made namely that the additionally selected points are also on the cylinder. If the assumption is true, the automatic process of identification can start, as was the case with the plane. Figure 6 shows the selected points after one and two successive iterating steps.

Recognising a cylinder usually requires more iterations than for a plane. The final result was obtained only after twelve steps. The selected points are shown in Figure 7. This example again stresses the need for a fast fitting algorithm. The fitting process required twelve iterations. During the last iteration, 3500 points were used.

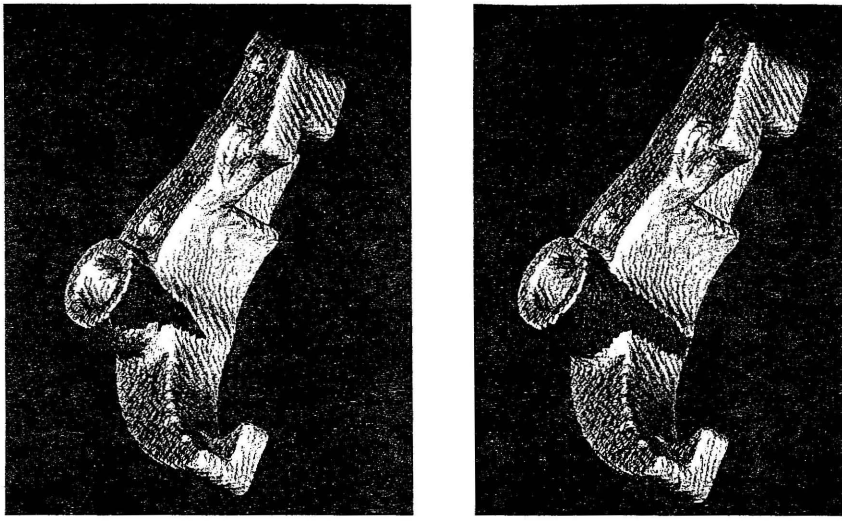

Figure6: The selected points after one and two steps

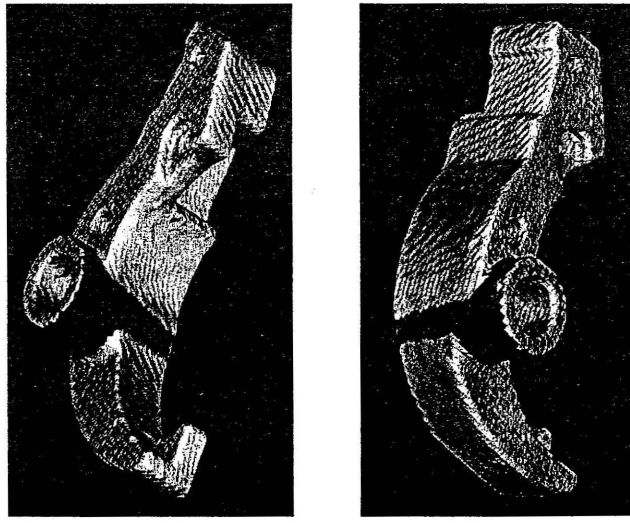

Figure7: Rear and front view of the resulting cylinder

The rear of the object clearly illustrates the effect of the statistical pitfall. The selected cylinder incorrectly includes a strip of the much larger cylinder that defines the complete rear area of the object. In reality, this strip does not belong to the selected cylinder. However, for this type of application the result is satisfactory and the problem purely academic.

6. EXTENDING THE METHOD

6.1 Full feature extraction

The ultimate tool for feature recognition is full feature extraction. For a full feature extraction solution, the input is a cloud of points and the output the same cloud but with all points belonging to an analytical feature removed and replaced by that feature. The feature recognition solution presented above does not have this functionality. It can however be used to implement such a functionality as shown in Figure 8.

In a first step, points are selected randomly. These must be points within the same neighbourhood of one another. In the STL-implementation, this could mean one or more adjacent triangles. The recognising algorithm is then applied for all supported entities. If one is found, the selected points are deleted and the least-squares entity is saved in a new list. If no entity is found, the selected points do not belong to any of the supported entities so they are removed from the original cloud and saved (as points) in the new list. The procedure is repeated until there are no more points in the cloud, at which time the new list will contain all the identified entities as well as a subset of the original points that could not be associated with any of the supported entities.

The algorithm of figure 8 was tested. Although it proved to be quite robust, it is too slow for practical applications. On files with a lot of free-form points that do not correspond to analytical entities, the algorithm is extremely slow, because it has to test all such points for each and every possible analytical object. If the point cloud is searched for the various analytical objects according to a pre-defined and logical sequence, the speed of the procedure can be greatly improved. The optimum sequence is to first try to identify and remove simple entities like planes and spheres from the point cloud. This should be followed by a search for more complex entities such as cylinders and cones, where after the very complex ones (e.g. torusses) could be identified manually. The foregoing procedure may seem a bit opportunistic, but it is practical and it does work.

It is important to note that the above procedure requires of the recognition algorithm to be able to detect whether an entity it is looking for is present or not (and as early as possible). This must be done for each supported entity separately and is not trivial.

The time-consuming fitting algorithm is placed within the most inner loop of the procedure. In order to achieve acceptable processing speeds, it is therefore of the utmost importance that this algorithm be optimised as far as possible. (See Gill [3] and Press [5])

6.2 On-line recognition

Until now, the described methods operate like a post-processor on point-clouds. From a co-ordinate-measuring machine control perspective, there is also a need for an on-line version of the solution. Presently, when an unknown or free-form object is measured, the common approach is to measure many points on the object's surface such that a good covering is obtained. Advanced measuring techniques use the local curvature of the surface of the object to determine how many sampling points are necessary for a required level of accuracy. When measuring an object containing analytical entities, this practice results in an unnecessary covering of the analytical entities. If such entities could already be detected during the measurement, the measuring path could be adapted for a dedicated entity measurement. In general, this will lead to a much shorter measuring time.

Unfortunately, the previous methods are not easily adaptable for on-line processing. At this moment, a complete solution for this problem does not exist.

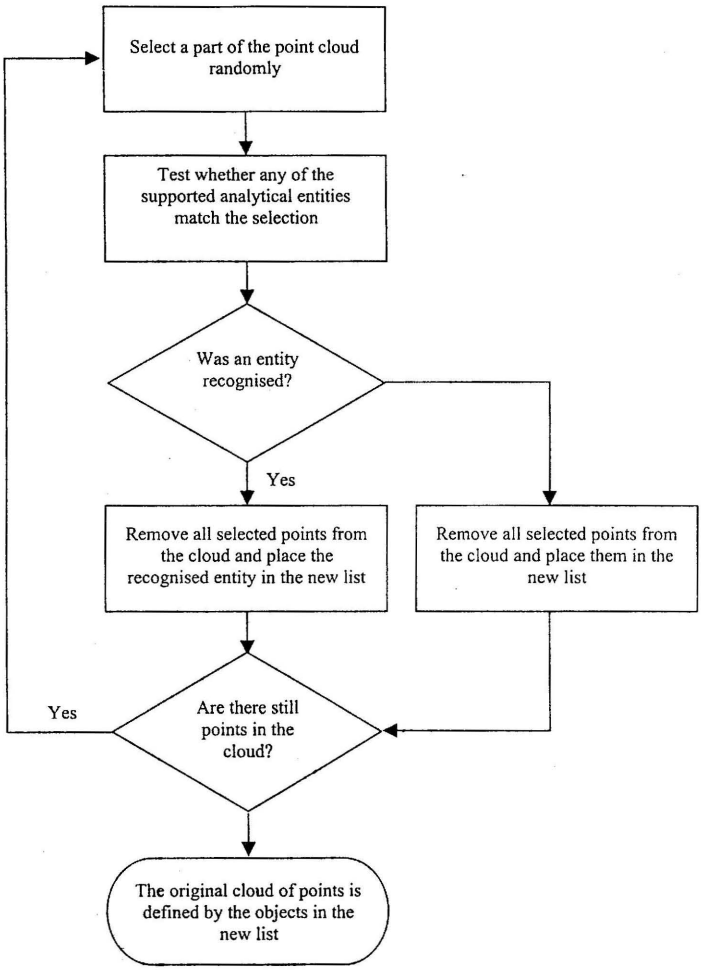

Figure 8: Full feature extraction algorithm

7. REFERENCES

[1] De Geeter J, 1998, "Constrained system state estimation and task-directed sensing", Ph.D. thesis, PMA KULeuven, Belgium.

[2] Forbes A B, February 1991, "Least-squares best-fit geometric elements", NPL Report DITC 140/98, Revised edition.

[3] Gill P E, Murray W and Wright H M, 1981, "Practical Optimization", Academic Press.

[4] Janssens M, 1998, "Sub-manifold theory leads to a significant reduction of the time complexity of fitting algorithms for least-squares geometrical entities", annex to Ph.D. thesis, PMA KULeuven, Belgium

[5] Press W H, Teukolsky S A, Vetterling W T and Flannery B P, 1992, "Numerical Recipes in C: The Art of Scientific Computing", 2nd Edition, Cambridge University Press.